Amendments to the Claims

This listing of claims will replace all prior versions, and listings of claims in the application.

1. (Currently Amended) A method for precise feedback data generation and updating during compile-time optimizations, within an optimizing compiler, comprising:

(1) accessing a first intermediate representation of source code of a computer program, wherein said first intermediate representation includes instructions instrumented into the source code of said computer program;

(2) annotating said first intermediate representation with previously-gathered frequency data from a plurality of sample executions of said computer program;

(3) updating said frequency data ~~according to a pre-defined propagation scheme~~ to maintain accuracy of said frequency data during compilation in a direction of increasing exactness;

(4) performing an optimization of said first intermediate representation annotated with said frequency data updated in step (3), thereby producing a transformed intermediate representation; and

(5) repeating steps (3) and (4) at least once during the same compilation pass.

2. (Original) The method of claim 1, wherein step (4) comprises the step of performing at least one of the following optimizations:

(i) dead code elimination;

(ii) dead store elimination;

(iii) branch elimination; and

(iv) code transformation.

3. (Original) The method of claim 1, wherein said first intermediate representation is a tree corresponding to a procedure within the source code of said computer program.

4. (Original) The method of claim 3, wherein step (2), comprises the steps of:

(a) constructing a control flow graph from said tree; and

(b) annotating a frequency value to an edge of said control flow graph, wherein said frequency value corresponds to the number of times that said edge was traversed during said plurality of sample executions of said computer program.

5. (Original) The method of claim 4, wherein said frequency value annotated to said edge of said control flow graph is one of the following:

(i) EXACT;

(ii) GUESS;

(iii) UNKNOWN;

(iv) UNINIT; and

(v) ERROR.

6. (Currently Amended) A computer program product comprising a computer usable medium having computer readable program code means embodied in said medium for causing an application program to execute on a computer that performs precise feedback data generation and updating during compile-time optimizations, within an optimizing compiler, said computer readable program code means comprising:

first computer readable program code means for causing the computer to access a first intermediate representation of source code of a computer program, wherein said first intermediate representation includes instructions instrumented into the source code of said computer program;

second computer readable program code means for causing the computer to annotate said first intermediate representation with previously-gathered frequency data from a plurality of sample executions of said computer program;

third computer readable program code means for causing the computer to update said frequency data ~~according to a pre-defined propagation scheme~~ to maintain accuracy of said frequency data during compilation in a direction of increasing exactness;

fourth computer readable program code means for causing the computer to perform an optimization of said first intermediate representation annotated with said frequency data updated by said third computer readable program code means, thereby producing a transformed intermediate representation; and

fifth computer readable program code means for causing the computer to re-execute said third and fourth computer readable program code means at least once during the same compilation pass.

9. (Currently Amended) A method for compile-time optimization comprising:

(1) accessing a first intermediate representation of source code of a computer program, wherein the first intermediate representation includes instructions instrumented into the source code;

(2) annotating the first intermediate representation with previously-gathered global and local frequency data from a plurality of sample executions of the computer program;

(3) updating the global and local frequency data ~~according to a pre-defined propagation scheme~~ to maintain accuracy of said frequency data during compilation in a direction of increasing exactness;

(4) performing an optimization of the first intermediate representation annotated with the global and local frequency data updated in step (3) to produce a transformed intermediate representation; and

(5) repeating steps (3) and (4) at least once during the same compilation pass.

10. (Previously Presented) The method of claim 9, wherein step (4) comprises the step of performing at least one of the following optimizations:

(i) dead code elimination;

(ii) dead store elimination;

(iii) branch elimination; and

(iv) code transformation.

11. (Previously Presented) The method of claim 9, wherein the first intermediate representation is a tree corresponding to a procedure within the source code.

12. (Previously Presented) The method of claim 11, wherein step (2) comprises the steps of:

(a) constructing a control flow graph from the tree; and

(b) annotating a global or local frequency value of the global and local frequency data to an edge of the control flow graph, wherein the global or local frequency value corresponds to the number of times that the edge was traversed during the plurality of sample executions of the computer program.

13. (Previously Presented) The method of claim 12, wherein the global and local frequency value annotated to the edge of the control flow graph is one of the following:

(i) EXACT;

(ii) GUESS;

(iii) UNKNOWN;

(iv) UNINIT; and

(v) ERROR.

14. (Currently Amended) A method for compile-time optimization comprising:

(1) accessing a first intermediate representation of source code of a computer program, wherein the first intermediate representation includes instructions instrumented into the source code;

(2) annotating the first intermediate representation with previously-gathered frequency data from a plurality of sample executions of the computer program;

(3) updating the frequency data ~~according to a pre-defined propagation scheme~~ to maintain accuracy of said frequency data in a direction of increasing exactness at multiple points during a compilation process;

(4) performing an optimization of the first intermediate representation annotated with the frequency data updated in step (3) to produce a transformed intermediate representation; and

(5) repeating steps (3) and (4) at least once during the same compilation pass.

15. (Previously Presented) The method of claim 14, wherein step (4) comprises the step of performing at least one of the following optimizations:

(i) dead code elimination;

(ii) dead store elimination;

(iii) branch elimination; and

(iv) code transformation.

16. (Previously Presented) The method of claim 14, wherein the first intermediate representation is a tree corresponding to a procedure within the source code.

17. (Previously Presented) The method of claim 14, wherein step (2) comprises the steps of:

(a) constructing a control flow graph from the tree; and

(b) annotating a global or local frequency value of the global and local frequency data to an edge of the control flow graph, wherein the global or local frequency value corresponds to the number of times that the edge was traversed during the plurality of sample executions of the computer program.

18. (Previously Presented) The method of claim 17, wherein the global or local frequency value annotated to the edge of the control flow graph is one of the following:

(i) EXACT;

(ii) GUESS;

(iii) UNKNOWN;

(iv) UNINIT; and

(v) ERROR.

19. (Previously Presented) The method of claim 17, wherein the global or local frequency value annotated to the edge of the control flow graph is one of the following:

(i) GUESS; and

(ii) UNKNOWN.

20. (Currently Amended) A method for compile-time optimization comprising the steps of:

(1) accessing a first intermediate representation of source code of a computer program, wherein the first intermediate representation includes instructions instrumented into the source code;

(2) annotating the first intermediate representation with previously-gathered estimated frequency data from a plurality of sample executions of the computer program;

(3) updating the estimated frequency data ~~according to a pre-defined propagation scheme~~ to maintain accuracy of said frequency data during compilation in a direction of increasing exactness;

(4) performing an optimization of the first intermediate representation annotated with the estimated frequency data updated in step (3) to produce a transformed intermediate representation; and

(5) repeating steps (3) and (4) at least once during the same compilation pass.

21. (New) The method of claim 1, wherein said frequency data comprises both inexact and exact values.

22. (New) The method of claim 1, wherein said updating step updates said frequency data from GUESS to EXACT values.

23. (New) The method of claim 1, wherein said updating step updates said frequency data from UNKNOWN to GUESS values.

24. (New) The method of claim 1, wherein said updating step updates said frequency data from UNINIT to GUESS values.

25. (New) The method of claim 1, wherein said optimization is performed in a direction of decreasing exactness.

26. (New) The method of claim 1, wherein said optimization is performed locally, and the updating is performed globally.

27. (New) The product of claim 6, wherein said frequency data comprises both inexact and exact values.

28. (New) The product of claim 6, wherein said updating step updates said frequency data from GUESS to EXACT values.

29. (New) The product of claim 6, wherein said updating step updates said frequency data from UNKNOWN to GUESS values.

30. (New) The product of claim 6, wherein said updating step updates said frequency data from UNINIT to GUESS values.

31. (New) The product of claim 6, wherein said optimization is performed in a direction of decreasing exactness.

32. (New) The product of claim 6, wherein said optimization is performed locally, and the updating is performed globally.

33. (New) The method of claim 9, wherein said frequency data comprises both inexact and exact values.

34. (New) The method of claim 9, wherein said updating step updates said frequency data from GUESS to EXACT values.

35. (New) The method of claim 9, wherein said updating step updates said frequency data from UNKNOWN to GUESS values.

36. (New) The method of claim 9, wherein said updating step updates said frequency data from UNINIT to GUESS values.

37. (New) The method of claim 9, wherein said frequency data comprises both inexact and exact values.

38. (New) The method of claim 9, wherein said optimization is performed in a direction of decreasing exactness.

39. (New) The method of claim 9, wherein said optimization is performed locally, and the updating is performed globally.

40. (New) The method of claim 14, wherein said updating step updates said frequency data from GUESS to EXACT values.

41. (New) The method of claim 14, wherein said updating step updates said frequency data from UNKNOWN to GUESS values.

42. (New) The method of claim 14, wherein said updating step updates said frequency data from UNINIT to GUESS values.

43. (New) The method of claim 14, wherein said optimization is performed in a direction of decreasing exactness.

44. (New) The method of claim 14, wherein said optimization is performed locally, and the updating is performed globally.